

Cyclic Scheduling Algorithm

Ravin Kumar

Department Of Computer Science,
Meerut Institute Of Engineering And Technology,
meerut-250005, Uttar Pradesh, India.
ravin.kumar.cs.2013@miet.ac.in

Abstract. CPU Scheduling have played a critical role in making efficient systems, It is the mechanism which allows one process to use the CPU for execution while other processes are put on hold because of unavailability any required resource, with the aim of maximizing the CPU utilization and reducing the waiting time and the turnaround time. This paper presents a new Scheduling Algorithm, which supports preemption, and is evaluated using turnaround time and response time. To show its effectiveness, its comparison is done with other traditional scheduling algorithms including First Come First Serve, Shortest Job First, and Round Robin Scheduling Algorithm, and as a result it was found that the proposed algorithm provides a new and effective scheduling approach. For the four test samples, the proposed algorithm reduces the average turnaround time compared with Round Robin Scheduling Algorithm with time quantum 3, while its average response time depends on the workload.
Keywords: Preemption, Scheduling, Resource Utilization, Response Time, Turnaround Time.

1 Introduction

Computers have become an important part of today's Information Age, due to their ability to perform several tasks simultaneously. Inside a computer, processors play a crucial role in decision making, and calculation related tasks. When processor have more than one process to execute, at that time Scheduling algorithms[1] are required to manage execution efficiency of processes. CPU Scheduling is one of the fundamental concepts of Operating System, which are designed with the aim of maximizing the utilization of available system resources, resulting in better, fair, and fast use of shared resources among the available processes. There are various traditional scheduling algorithms which were designed with the aim of better resource utilization, some of them are FCFS, SJF, and Round Robin scheduling algorithm[2]. There are many factors that require attention during scheduling process, this includes burst time, arrival time, and priority of the process, with the aim of reducing the average waiting time and average turnaround time [3]. We have studied various scheduling algorithms from existing literature.

FCFS (First Come First Serve)- This is one of the most earlier approach which was used for scheduling purpose. This follows the ideology of "First Come First Serve"

i.e. assigning CPU time first to that process, which requested them first. [4].

SJF (Shortest Job First) – In this approach, CPU is allocated to that process which have the smallest burst time available, i.e. the job with the shortest computation time executed first. When a job first arrives, it is put inside the ready queue, and then the job which have smallest burst time is chosen for execution [5].

Round Robin Scheduling – In this approach, a preemptive time slicing is performed with the help of a time quantum or time slice. A time slice or time quantum is a small unit of time, due to which it can give the effect of all processor sharing the CPU equally with its given time quantum.

This paper proposes a new CPU scheduling algorithm, which can be used for the same purpose, and when compared with other approaches, it shows comparatively good results.

2 Related Work

Kumar M. et al.[6] designed a method to suggest the length of next CPU burst in SJF, and showed that approximate length of next CPU burst is similar to length of previous request. Khan R. et al.[7] performed a comparative study over SJF and FCFS for similar priority jobs. Li J. et al.[8] improvised the FIFO algorithm by using fuzzy logics, and showed that it reduces the execution time for tasks and increases the resource utilization. Indusree R. et al.[9] suggested an improvement to RR by calculating the dynamic time quantum each burst time for processes present in ready queue. Singh J. et al.[10] developed a multi queue based scheduling approach for cloud architectures. Lin Z. et al.[11] showed that weighted RR performs better than FCFS in Hama architecture. Jha H. et al.[12] proposed a hybrid algorithm of RR and priority scheduling, which was an improvement over both traditional RR. In servers based on OpenFlow, Peng R. et al.[13] suggested a weighted RR algorithm that can be used for load balancing. Chaturvedi P. et al.[14] utilized improved RR in workflow applications in cloud. Rao G.S.N. et al.[15] proposed a dynamic time calculation mechanism for RR for improving its performance.

3 Proposed Scheduling Algorithm

Let us assume a set of processes $P_0, P_1, P_2, \dots, P_n$ that are needed to be executed using the proposed scheduling algorithm. QUEUE is representing a queue type data structure which holds those process which have been executed partially. Now, let us assume that there is a set 'L' which hold all those process are not completed and their Arrival Time is either less than or equal to the current time. Flow chart of the proposed algorithm is given in Fig. 1.

Cyclic Scheduling Algorithm ()

Parameters Q_{OLD} and Q_{NEW} represents the dynamic time quantum, CNT represents an integral value that is used to decide the dynamic time quantum. RT_x is the remaining time of the selected process. Max and Min are maximum and minimum remaining time among processes that are either present in L or are in QUEUE. N represents total number of active and arrived processes.

BEGIN

Step 1: Set $L = \{ \}$, $QUEUE = \text{Null}$, $CNT = 0$ and $Q_{NEW} = 0$.

Step 2: Add all the new process that had arrived, in set L , and update $CNT = CNT + 1$

Step 3: $P_X = \text{Select_Process}(CNT)$

Step 4: IF number of process in $L == 1$

then

IF $QUEUE == \text{Null}$

then

$Q_{OLD} = Q_{NEW} = 1$

ELSE

$Q_{NEW} = Q_{OLD}$

end IF

ELSE

// calculating new time quantum.

$Q_{NEW} = \text{minimum} (RTX, (Max - Min) / 2, \Sigma RT / N)$

$Q_{OLD} = Q_{NEW}$.

// in above formula remove all zero values, and calculate $Q_{NEW} (\geq 1)$.

end IF

Step 5: After running P_X for Q_{NEW} time, remove P_X from L , and add (P_X, N_X) to $QUEUE$, here N_X can be calculated as :

$N_X = N_{TEMP} + \text{number of } (P, N) \text{ pairs present in } QUEUE.$

Where,

$N_{TEMP} = \text{Number of process in } L + \text{Number of process that just arrived during this execution.} + 1$

Step 6: Calculate_Jump ()

Step 7: IF All Processes are completed and $L = \{ \}$ and $QUEUE == \text{Null}$

then

Exit.

Else

Goto step 2.

END

Select_Process (CNT)

This function is used to decide which process to be chosen for execution from set L .

BEGIN

Step 1: IF $CNT \% 2 == 1$

then

return Process which have minimum remaining time in set L .

ELSE

return Process present in L and have arrival time closer to
(Maximum Arrival time + Minimum Arrival time) / 2.

// Maximum and Minimum are calculated among processes present in L .

END

If more than one process satisfies `Select_Process`, the process with the earliest arrival time is selected; if arrival times are also equal, then the process with the lower identifier is selected. Fractional values of Q_{NEW} are retained. During `Calculate_Jump`, after an entry is removed, the shifted entry at the same position is evaluated next.

`Calculate_Jump ( )`

This function is used to make use of dynamic time quantum to place processes back from `QUEUE` to set `L`.

`BEGIN`

Step 1: For each (P_T, N_T) present in `QUEUE`

 IF $N_T \leq$ Number of (P, N) pairs present in `QUEUE`.

 Then

 For each (P_Y, N_Y) present in `QUEUE`

$N_Y = N_Y - 1$

 end For

 Remove (P_T, N_T) from `QUEUE`.

 IF Remaining time of $P_T > 0$

 then

 Put process P_T in set `L`.

 end IF

 end IF

 end For

`END`

Given below is the flow chart of the proposed scheduling algorithm (Fig. 1).

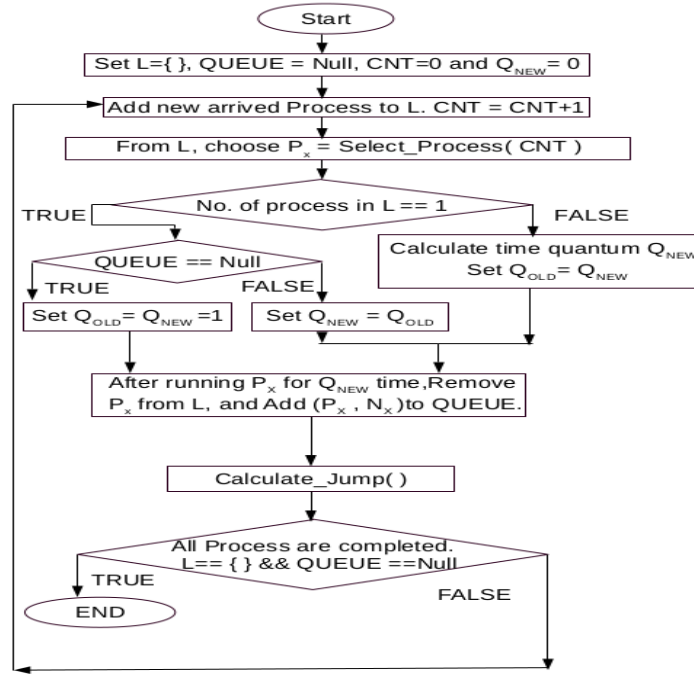


Fig. 1. Flow chart of Cyclic Scheduling Algorithm.

4 Working Demonstration

To better understand the proposed algorithm we have considered a set of processes with burst and arrival times, so that the behavior of the new proposed scheduling algorithm can be easily understood (Table 1).

Table 1. Sample Data to demonstrate the procedure of Cyclic Scheduling Algorithm.

Process	Arrival Time	Burst Time
P0	0	2
P1	0	3
P2	2	1

Initially we have $L = \{ \}$, $QUEUE = \text{Null}$, $CNT = 0$ and $Q_{\text{NEW}} = 0$.

Table 2 Shows the status of processes, after applying the proposed CSA.

Table 2. Applying Cyclic Scheduling Algorithm.

Current Time	CNT	QNEW	Set L { }	QUEUE
0	0	0	{ P0, P1 }	Null
1	1	1	{ P1 }	(P0 , 2)
2	2	1	{ P0, P2 }	(P1 , 2)
3	3	1	{ P1, P2 }	(P0 , 2)
4	4	1	{ P2 }	(P1 , 2)
5	5	1	{ P1 }	Null
6	6	1	{ }	Null

After applying the proposed scheduling algorithm, the response time and turnaround time of each process are evaluated, it is given in Fig. 2.

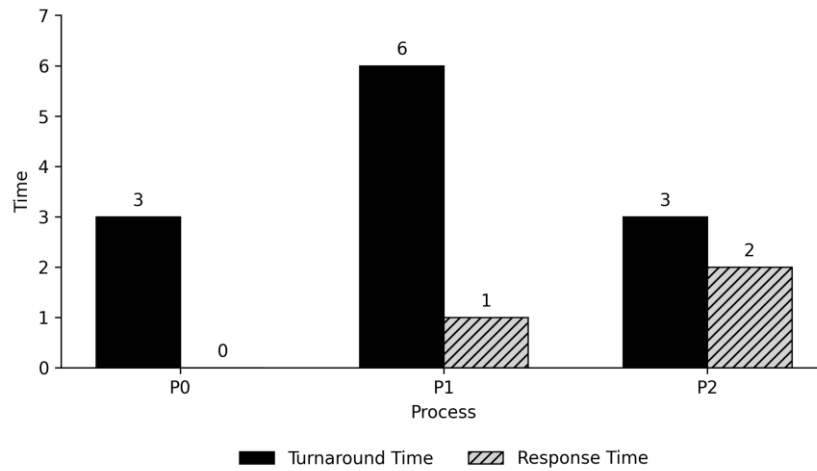


Fig. 2. Representation of individual turnaround and response time using Cyclic Scheduling Algorithm.

Response time is the interval from process arrival to its first CPU allocation, and turnaround time is completion time minus arrival time. Using these definitions, the average values for the sample are shown in Table 3.

Table 3. Overall performance of the proposed scheduling algorithm on sample data.

Parameter	Value
Average Turnaround Time	4.00
Average Response Time	1.00

5 Result And Discussion

In the proposed algorithm, the outer loop containing set 'L' and queue 'QUEUE' makes sure that the algorithm have a cyclic nature for the execution of process, similar to that of round robin algorithm, But instead of using the order in which the processes first arrived, Our algorithm have an inner mechanism which helps in improving the algorithm's performance by providing dynamic time quantum, and then selecting a process on the basis of arrival time and remaining time to execute in the outer loop.

Cyclic Scheduling Algorithm is tested with four test data sets and compared with FCFS, non-preemptive SJF, and Round Robin with time quantum 3. The comparison reports average turnaround time and average response time.

Table 4. Sample 1- Burst time in increasing order.

Process	Arrival Time	Burst Time
P0	0	2
P1	1	7
P2	2	8
P3	3	14

In Fig. 3, average turnaround time and average response time obtained for Table 4 after applying each algorithm.

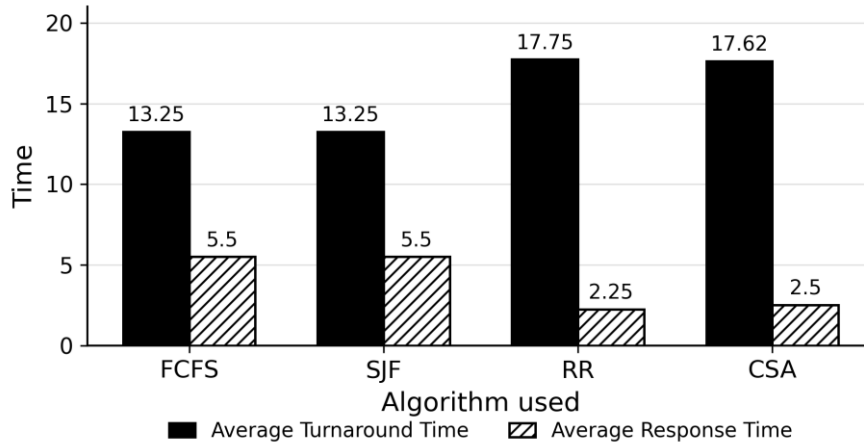


Fig. 3. Comparison of average turnaround time and average response time of FCFS, SJF, RR and CSA scheduling algorithms.

Let us consider another sample of data to further check the result, Given below is the set of assumed data on which proposed scheduling algorithm is applied and then the comparison are done with other algorithms (Table 5).

Table 5. Sample 2- Burst time in decreasing order.

Process	Arrival Time	Burst Time
P0	0	14
P1	1	8
P2	2	7
P3	3	2

Given below is the graphical representation of the average turnaround time and average response time obtained after applying each algorithm (Fig. 4).

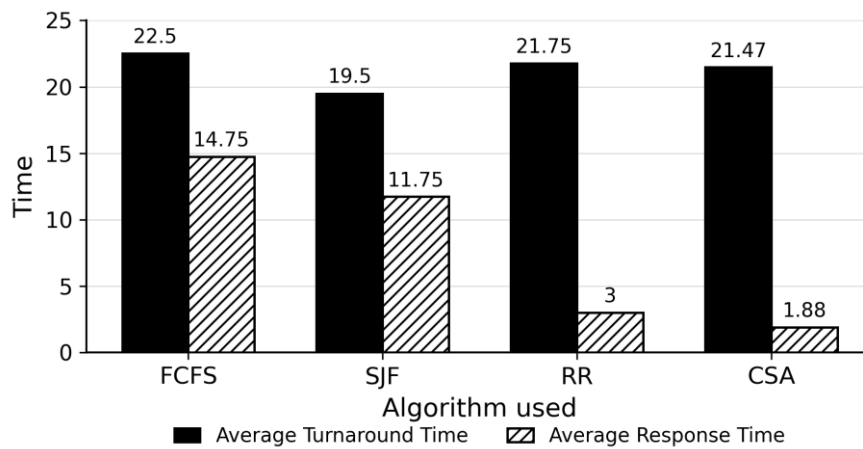


Fig. 4. Comparison of average turnaround time and average response time of various scheduling algorithms.

Similarly, consider another sample of data on which we first apply the proposed scheduling algorithm, and then a comparison is done with other algorithms (Table 6).

Table 6. Sample 3- Burst time in random order.

Process	Arrival Time	Burst Time
P0	0	2
P1	1	8
P2	2	7
P3	3	14

For the above Sample, the comparative analysis of performance obtained after implementing different scheduling algorithms is given in Fig. 5.

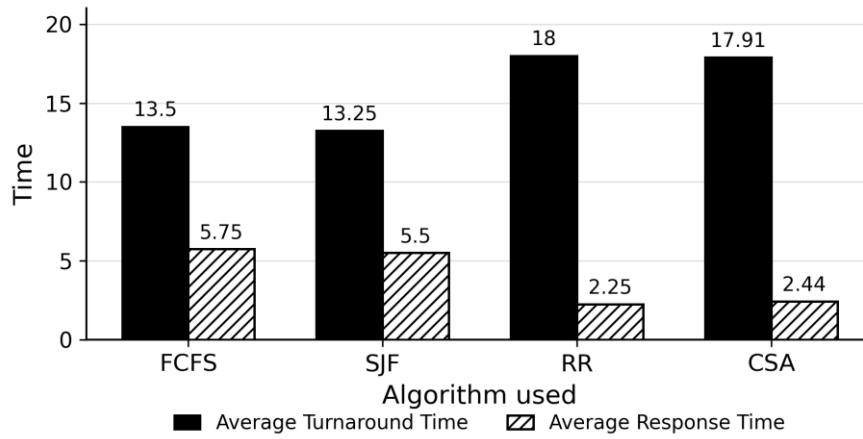


Fig. 5. Comparison of average turnaround time and average response time of various scheduling algorithms.

The fourth sample is generated by randomly assigning arrival and burst time to a set of ten processes (Table 7).

Table 7. Sample 4- Set of ten processes assigned with random values.

Process	Arrival Time	Burst Time
P0	0	3
P1	1	14
P2	2	7
P3	2	2
P4	3	11
P5	3	5
P6	4	8
P7	5	13
P8	6	9
P9	6	3

Given below is the comparative analysis of different scheduling algorithms over the above set of processes (Fig. 6).

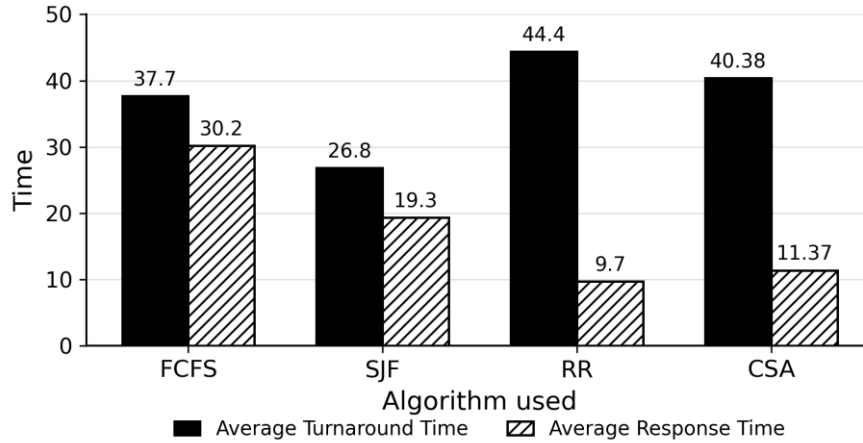


Fig. 6. Comparison of average turnaround time and average response time of various scheduling algorithms.

6 Conclusion

In this paper, an advanced method is introduced for scheduling a set of processes having the same priority. For the four test samples, the proposed algorithm reduces the average turnaround time compared with Round Robin Scheduling Algorithm with time quantum 3, while its average response time depends on the workload. This algorithm provides a new way of scheduling processes, and also opens a way for developing new hybrid scheduling algorithms by combining the proposed algorithm with other traditional algorithms. Future research work can be done on adding priority mechanism to the proposed algorithm.

References

1. Operating System Concepts 8th Edition, Silberschatz Galvin Gagne, WILEY.
2. Neha Mittal, Kushbu Garg, Sshish America, A Paper on Modified Round Robin Algorithm, IJLTEMAS, Volume IV, Issue X1, 2005, ISSN 2278-2540.
3. Seltzer, M.P. Chen and J outerhout, 1990. Disk scheduling revisited in USENIX. Winter technical conference. Shamim H M 1998. Operating system, DCSA-2302.
4. Raman, Dr. Pradeep Kumar Mittal, An Efficient Dynamic Round Robin CPU Scheduling Algorithm, IJARCSSE, 5 May 2014, ISBN: 2277 128X.
5. Weiming Tong and Jing Zhao, "Quantum Varying Deficit Round Robin Scheduling Over Priority Queues", International Conference on Computational Intelligence and Security, pp. 252-256, China, 2007.
6. M. R Mahesh Kumar, B Renuka Rajendra, "Prediction of Length of the Next CPU Burst in SJF Scheduling Algorithm using Dual Simplex Method", 2014 2nd International Conference on current Trends in Engineering and Technology ICCTET', 2014.
7. Khan R, Kakhani G (2015) Analysis of priority scheduling algorithm on the basis of fcfs and sjf for similar priority jobs. Int J Comput Sci Mob Comput 4:324–331
8. Li, J., Ma, T., Tang, M., Shen, W., Jin, Y., 2017. Improved FIFO scheduling algorithm based on fuzzy clustering in cloud computing. Information 8(1), p.25

9. Indusree, J.R. and Prabadevi, B., 2017, November. Enhanced round robin CPU scheduling with burst time based time quantum. In IOP Conference Series: Materials Science and Engineering (Vol. 263, No. 4, p. 042038). IOP Publishing.
10. Singh, J. and Gupta, D., 2017. An Smarter Multi Queue Job Scheduling Policy for Cloud Computing. International Journal of Applied Engineering Research, 12(9), pp.1929-1934.
11. Zhijie, L. I. N., and Haifeng HUANG. "Research on Weighted Rotation Fair Scheduling Algorithm Based on Hama Parallel Computing Framework" DEStech Transactions on Engineering and Technology Research apop (2017).
12. Jha, H., Chowdhury, S. and Ramya, G., 2017. Survey on various Scheduling Algorithms. Imperial Journal of Interdisciplinary Research, 3(5).
13. Peng, R. and Ding, L., 2017, August. Research on virtual network load balancing based on OpenFlow. In AIP Conference Proceedings (Vol. 1864, No. 1, p. 020014). AIP Publishing.
14. Chaturvedi, P. and Sharma, S., 2017. An Provision For Workflow Scheduling using Round Robin Algorithm in Cloud. International Journal of Advanced Research in Computer Science, 8(5).
15. Rao, G.S.N., Srinivasu, N., Srinivasu, S.V.N. and Rao, G.R.K., 2015. Dynamic Time Slice Calculation for Round Robin Process Scheduling Using NOC. International Journal of Electrical and Computer Engineering, 5(6)